F

Chapitre 2

Enjeux des systèmes à criticité multiple sur processeurs multicœurs

Sommaire

La conception des systèmes embarqués, typiquement automobiles, a subi de fortes évolutions orientées vers de nouvelles fonctionnalités centrées sur le logiciel. Ces évolutions demandent des capacités de calcul de plus en plus importantes et donc des architectures matérielles pour supporter la demande grandissante en fonctionnalités. Par ailleurs le contexte industriel mène à la disparition des calculateurs d'antan, monocœurs, pour se focaliser sur des calculateurs plus complexes et puissants, multicœurs. Cette tendance au multicœur provient à la fois d'une limitation technologique et d'un besoin grandissant : la façon d'augmenter les capacités de calculs par les méthodes classiques (montée en fréquence) atteint ses limites et les capacités d'exécution concourante de logiciel est de plus en plus demandée dans un contexte aux contraintes financières et de time-to-market fortes. C'est ainsi que né la volonté de passer sur des architectures électriques et électroniques plus centralisées via l'utilisation d'une quantité d'unités de calcul réduite, mais intégrant un plus grand nombre de fonctionnalités en leur sein. Cette volonté implique cependant une superposition des difficultés inhérentes aux architectures matérielles plus complexes

avec les contraintes de sûreté de fonctionnement du logiciel. Nous faisons donc face à des systèmes à criticité mixte exécutés sur des calculateurs aux mécanismes complexes. Nous verrons ainsi dans ce chapitre quels sont les aspects essentiels de ce contexte et ses spécificités à prendre en compte pour proposer de nouveaux éléments de réponse dans la conception de systèmes à criticité mixte sur processeurs multicœurs. Nous conclurons cette partie avec la présentation de la problématique à laquelle on tentera de répondre par la suite ainsi que la présentation des différents chapitres de contribution de cette thèse.

2.1 Évolutions des Systèmes embarqués

2.1.1 Nouveaux systèmes intelligents et connectés

Si l'on prend l'exemple du domaine automobile, depuis près de 30 ans l'industrie n'a cessé de faire évoluer la façon de concevoir les véhicules et notamment leurs systèmes sous-jacents. Comme illustré avec le diagramme 2.1, la transition s'est faite de modifications purement mécaniques vers des évolutions électriques, puis électroniques et de plus en plus intelligentes. Les dates indiquées ici étant choisies de façon arbitraire, selon les grandes dates qui ont pu marquer le domaine. Les voitures se sont modernisées avec l'ajout de calculateurs dédiés à des fonctions internes ou des services. Le développement des technologies de l'industrie 4.0 mène à une augmentation exponentielle du logiciel embarqué dans l'automobile au cours des 15 dernières années [**?**], avec la présence de plus de 60 calculateurs embarqués dans certains modèles. Les contrôles mécaniques et autres systèmes électriques "simples" cèdent la place au monde du numérique. Les équipements électroniques et logiciels se multiplient au sein du véhicule pour l'aide à la conduite (*ADAS*) et l'ajout de services [**?**]. Test `[U+202F]` ! Ainsi, du simple Système Anti-blocage

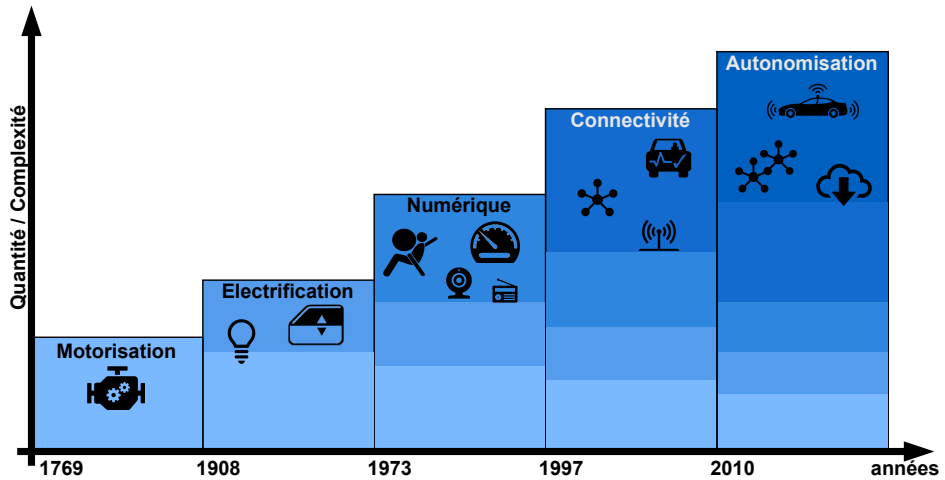

FIGURE 2.1 – Principaux domaines d'évolution des systèmes automobiles au fil du temps

des roues (ABS), on a introduit des Assistants à la Conduite tels que le Freinage

d'Urgence (*Emergency Braking System*) ou encore le Système de Gestion de Ligne (*Lane Support System*) qui permet à la fois l'Avertissement de Dépassement de Ligne (*Lane Departure Warning*), l'Assistant de Maintien de Ligne (*Lane Keeping Assist*) et le Maintien de Ligne d'Urgence (*Emergency Lane Keeping*)... et il ne s'agit là que de 2 fonctionnalités supplémentaires ! En parallèle, la voiture qui devient de plus en plus automatisée, voire autonome. Elle gagne en connectivité avec la prise en compte de données extérieures et un lien direct au cloud pour proposer une diversité de services : météo, divertissement, trafic routier, pour n'en citer que quelques exemples. Les systèmes embarqués deviennent par conséquent aussi connectés. On parle de communications *car-to-car* entre véhicules ou *car-to-infrastructure* entre véhicule et infrastructures routières par exemple. Cette ouverture du système à son environnement est à double tranchant. D'une part cela offre de nouveaux horizons de fonctionnalités et optimisations de conduite, avec des possibilités d'évolutivité simplifiée. Mais d'autre part la complexité va grandissante avec les enjeux d'ingénierie que cela implique.

De façon plus générale, le contexte industriel actuel fait émerger de nouvelles technologies basées sur des logiciels de plus en plus complexes et performants. Cela est rendu possible via l'émergence d'architectures matérielles toujours plus puissantes et performantes. Ces améliorations permettent le développement et la mise en application de nouvelles technologies comme les réseaux de communication sans fil haute performance ou encore l'usage d'intelligences artificielles. On retrouve ainsi un nombre grandissant de fonctionnalités directement embarquées dans l'automobile, l'avion, le train pour répondre à la fois à de nouveaux besoins fonctionnels : assistance à la conduite/pilotage, tableaux de bord, etc. et à des besoins de confort d'usage : info-divertissement, connectivité, automatisations...

D'un point de vue logiciel, les mises à jour de systèmes embarqués incluent à la fois de nouvelles fonctionnalités critiques pour le bon fonctionnement du système, mais aussi des ajouts moins critiques. Ces mises à jour de services non essentiels amplifient la multiplicité des niveaux de criticités du logiciel embarqué et donc la cohabitation entre sous-systèmes critiques et sous-systèmes non-critiques que l'on pourrait qualifier de "confort".

D'un point de vue matériel, il y a de fortes convergences sur les architectures employées dans les différents domaines. Historiquement, on retrouvait en premier lieu des calculateurs monocœurs. Cependant, les diverses évolutions exigences ont fait apparaître les limites en capacité de calcul de ces derniers. La montée en fréquence de fonctionnement atteint ses limites à cause de la chauffe et la consommation que cela implique. Tandis que l'augmentation du nombre de transistors qui composent les processeurs arrive aux abords des limites physiques : la taille d'une cellule logique arrive aux mêmes dimensions que les atomes de silicium dont il est composé.. ! Pour ces raisons, la notion de calculateur multicœur est apparue dès les années 1950 [**?**] où les fondeurs s'orientent vers des processeurs où la montée en puissance est assurée par la multiplication des unités de calcul (dit "cœurs") parallèles dans le processeur. On passe ainsi de monocœurs aux duals/quadri cœurs... et l'on va aujourd'hui jusqu'à des "monstres" de puissance à plus de 128 cœurs. Et cette évolution est la bienvenue dans tous les secteurs concernés, allant du grand public dans les ordinateurs, téléphones et autres multimédias jusqu'aux applications industrielles

en passant par les usages de serveurs réseaux et centres de calculs.

Il existe divers types d'architectures matérielles dans l'ensemble des évolutions multicœurs que l'on retrouve aujourd'hui. On pourrait de façon simple différencier entre les multicœurs classiques, les manycœurs et à l'extrême ce que l'on connaît sous le nom de GPU, les processeurs graphiques.

Multicœurs "classiques"

Basés sur le cache

Basés sur scratchpad (SPM) Dans le cadre du contexte automobile, on se dirige de cette façon vers un nouveau paradigme, où la voiture n'est plus un système mécanique sur lequel on adjoint du logiciel, mais à l'inverse un superordinateur multifonctionnel auquel on implante des roues et un moteur. Les systèmes automobiles sont ainsi devenus des systèmes cyberphysiques qui entrent en interaction à la fois avec les utilisateurs et l'environnement. On distingue deux grands domaines de logiciels embarqués dans le véhicule. Tout d'abord *l'info-divertissement*, qui réunit les systèmes multimédias et autres affichages non nécessaires à l'usage primaire du véhicule. Et deuxièmement les calculateurs enfouis qui réalisent des fonctions essentielles qui ne sont pas nécessairement visibles de l'utilisateur, telles que le contrôle moteur.Cette disruption apporte de nouveaux enjeux, notamment de sécurité, vie privée, mais aussi sur la prédictibilité et sûreté de fonctionnement du système grâce à sa complexification. Cela fait donc évoluer les systèmes embarqués dans un environnement profondément à risques, mais qui en plus s'accompagne de contraintes fortes.

2.1.2 Enjeux industriels et économiques

Historiquement, les calculateurs embarqués étaient conçus de manière *ad hoc* : le logiciel et le matériel étaient intimement liés. Cela conduit à un nombre de calculateurs très important, chaque calculateur apportant une fonctionnalité qui lui est propre. On se retrouve face à des architectures avec un grand nombre d'unités de calcul interconnectées. Dans les systèmes automobiles, on peut noter trois principales propriétés dans ces ECU (*Electronic Control Unit*) :

— *interconnectés*,

— les applications et services sont intégrés dans des sous-systèmes complexes (*Software Components*),

— les fonctions sont *distribuées* sur plusieurs calculateurs.

Ce type d'architecture distribuée a des inconvénients évidents en terme d'évolutivité du système et coût de développement. A chaque changement de support physique le logiciel doit passer par un nouveau stade de développement plus ou moins conséquent. Inversement, une mise à jour du logiciel ou un ajout de fonctionnalité va demander une prise en compte de l'intégration matérielle avec potentiellement des modifications matérielles pour suivre les évolutions. Chaque ajout de fonctionnalités

va de cette façon ajouter de nouveaux calculateurs dédiés, complexifiant d'autant plus l'architecture.

On pourra mentionner aussi des contraintes d'encombrement. Les systèmes embarqués ont une forte tendance à la miniaturisation pour des raisons diverses selon les domaines. Cela permet une réduction de poids, essentiel pour tous les systèmes volants (avions, drones...) mais aussi d'encombrement pour des domaines comme l'automobile ou le ferroviaire qui doivent en toute circonstance rester dans des dimensions standards. Cette contrainte se fait beaucoup sentir avec l'arrivée des voitures autonomes par exemple, où les premiers prototypes – bien que fonctionnels – se sont avérés trop chargés et encombrants avec le surplus d'équipement pour être transposables facilement en produits commercialisables tel-quel.

Au regard de ces enjeux, l'évolution future naturelle est de réduire le nombre de calculateurs embarqué, en passant d'un grand nombre d'unités de calcul à une quantité limitée de supercalculateurs", qui vont agréger différentes tâches. On passe de cette façon d'un système distribué à un système fédéré basé sur des calculateurs primaires accompagnés de processeurs satellites qui gèrent le strict nécessaire à hauteur des différents capteurs/actionneurs. Cela permet de réduire les coûts et l'encombrement, qui va diminuer par la même occasion la quantité de câblages requis. Ce type d'architecture va faciliter l'évolutivité qui sera donc bien plus axée sur des mises à jour logicielles sans toucher au matériel. La connectivité permettant le concept du véhicule *"as-a-service"*, qui va pouvoir évoluer et se mettre à jour régulièrement à distance (*Over-the-Air Updates*).

2.1.3 Calculateurs multicœurs

Calculateurs Multi-cœurs La première évolution des calculateurs a été naturellement d'augmenter leur fréquence de fonctionnement et donc le nombre d'instructions par unité de temps réalisable. Ceci étant dit cette méthode a présenté ses limites, avec l'augmentation proportionnelle de la chauffe du processeur et une plus forte sensibilité aux perturbations. C'est pourquoi on a alors cherché à augmenter les capacités de parallélisme, pour améliorer l'efficacité. La notion de calculateur multicœur est apparue dès les années 1950 [**?**] pour nous amener aux architectures physiques actuelles. Le principe est de disposer d'un plus grand nombre d'unités de calculs (dit cœurs) qui pourront ainsi exécuter des instructions en parallèle. Dans le cas d'une parallélisation au niveau circuit (*chip-level multiprocessing - CMP*), plusieurs cœurs sont intégrés au sein d'un même boîtier. La mémoire locale est alors partagée à différents degrés entre les cœurs. De façon à décongestionner les accès mémoires et accélérer ces dernières, une hiérarchie mémoire est mise en place, associant des espaces mémoires progressivement plus petits et rapides en fonction de leur proximité au processeur. Il s'agit ici de trouver un équilibre entre coût de la mémoire et vitesse d'accès aux données. En effet, cette dernière dispose de trois caractéristiques antagonistes :

— la **latence** - temps d'accès aux données,

— la **bande passante** - débit de données accessible,

— la **taille** mémoire - espace mémoire disponible (pour un coût donné).

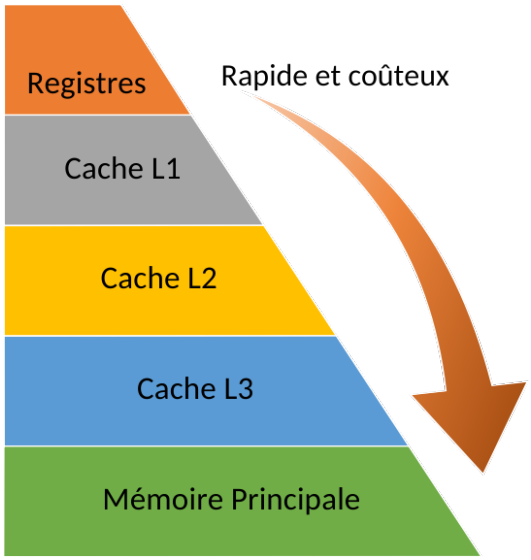

FIGURE 2.2 – Schéma des types de mémoire intégrées selon leur taille et coût

Un espace mémoire pourra donc soit être de petite taille, mais rapide au niveau de son temps d'accès, soit de grande taille et plus lent comme schématisé dans la 2.2. On a par conséquent au plus proche des cœurs les registres, de taille très limitée (octets) mais au temps d'accès très rapide : ils sont la base pour toutes les opérations effectuées par le processeur. À l'opposé, la mémoire principale, de très grande taille (Go/To) pour laquelle tous les cœurs doivent passer par un bus commun pour y accéder. C'est donc la mémoire la plus lente d'accès mais aussi la moins coûteuse.

Plusieurs intermédiaires ont été mis en place entre ces deux types de mémoire. Il s'agit typiquement de trois niveaux de cache L1, L2 et L3. Les caches L1 et L2 sont dits non partagés, c'est-à-dire propres à chaque cœur. Le cache L3 est partagé entre les cœurs. Ce dernier niveau est classiquement appelé LLC (*Last Level Cache*) et donne la limite entre les espaces mémoire limités en cache avec des accès rapides d'une part et la mémoire principale qui va provoquer de grands ralentissements d'autre part. La gestion du contenu de ces caches (en lecture et écriture) est géré par une politique d'accès mémoire. Cette politique est essentielle à un usage efficace des caches du fait de leur espace limité qui demande à faire des choix sur son usage. Cela est peu documenté par les constructeurs, et chacun aura sa façon de faire.

La méthode de base la plus répandue étant empirique, par principe de localité temporelle [**?**] et spatiale [**?**]. On considère que plus une donnée a été récemment accédée, plus elle a de chance d'être à nouveau utilisée. De même si une donnée est sollicitée, alors les données proches spatialement ont aussi plus de chance d'être utilisées. Nous n'iront pas plus dans les détails sur les politiques de gestion d'accès à la mémoire. Il faut garder en mémoire qu'elle est plutôt subie par les industriels qui intègrent le matériel dans leurs systèmes. Pour un processeur donné on aura certaines performances de calcul et accès mémoire, et il faudra mettre en comparaison les performances "par défaut" d'un logiciel sur une architecture donnée face au même système, mais avec des surcouches de gestion du logiciel apportées par l'intégrateur.

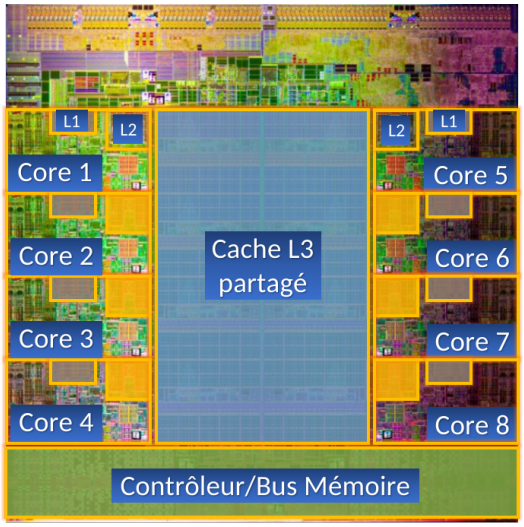

FIGURE 2.3 – Exemple d'architecture multicœur - Processeur Intel I7

Un exemple de processeur multicœur avec hiérarchie mémoire est indiqué en figure 2.3. On y retrouve plusieurs cœurs, ayant chacun leur propre niveau de cache L1 et L2. Le Niveau de cache L3 étant partagé entre tous les cœurs.

2.2 Risques et difficultés matérielles

2.2.1 Ressources partagées et Interférences

2.2.2 Conséquences logicielles

2.2.3 Constat et Objectifs

2.3 Contraintes et Hypothèses

2.3.1 Standards industriels

2.3.2 Contraintes d'intégration

De façon plus générale, les enjeux industriels peuvent varier selon les domaines. Ceci étant dit on peut nommer points principaux, qui sont ceux que l'on va tenter de prendre en considération dans cette étude. La première d'entre elle est l'imposition de capacités de déploiement rapides (*"Time-to-market"* réduit). Les itérations entre générations demandent des coûts de développement les moins importants possibles. Cela permet des cycles courts et réactifs qui s'adaptent aux évolutions technologiques. Cette contrainte industrielle est structurante sur les choix de conception, ce qui nous ramène souvent au principe "KISS", i.e. "Keep It Safe and Simple" dans notre cas. C'est une philosophie que j'ai souhaité maintenir le long de cette thèse afin de tenter une approche un peu différente des principales recherches actuelles qui tentent souvent d'aller dans des niveaux de détails toujours plus précis et complexes pour répondre aux difficultés technologiques. Comme on le verra plus tard, il existe ainsi

des solutions très sophistiquées qui donnent de bons résultats théoriques, mais qui ne se sont pas généralisés dans un contexte industriel. Les question de complexité d'implémentation et simplicité de maintenance dans un cas réel semblent donc relativement déterminantes pour mesurer la pertinence d'une nouvelle contribution à la sûreté des systèmes embarqués.

2.4 Grandes approches du domaine

2.4.1 Mécanismes de contrôle

2.4.2 Mécanismes de réaction

2.5 Contribution de la thèse et objectifs

Bibliographie

[Blanchet 2016] Blanchet, M. *Industrie 4.0 : nouvelle donne industrielle, nouveau modèle économique*. Géoéconomie, vol. 82, no. 5, page 37, 2016. (Non cité.)

[Durrieu 2014] Durrieu, G., Faugere, M., Girbal, S., Pérez, D. G., Pagetti, C. et Puffitsch, W. *Predictable Flight Management System Implementation on a Multicore Processor*. Dans Embedded Real Time Software (ERTS'14), 2014. (Non cité.)

[Schmidt 2010] Schmidt, A., Dey, A. K., Kun, A. L. et Spiessl, W. *Automotive User Interfaces : Human Computer Interaction in the Car*. Dans Extended Abstracts on Human Factors in Computing Systems, pages 3177–3180. ACM, 2010. (Non cité.)

[Smotherman 2005] Smotherman, M. *History of Multithreading*. Retrieved on, pages 12–19, 2005. (Non cité.)

[Wilkes 1965] Wilkes, M. V. *Slave Memories and Dynamic Storage Allocation*. IEEE Transactions on Electronic Computers, no. 2, pages 270–271, 1965. (Non cité.)